REMARKS

Claims 1-5, 7, 11-14 and 20-30 were previously pending in the present application. Please cancel claims 5, 7, and 14 and add claims 31-39. No new matter was added to the specification as a result of adding these claims.

The following remarks address the rejections of claims 1-4, 11-13 and 20-30 as set out by Examiner in the Final Office Action mailed on April 22, 2005.

Rejections of Claims 1 and 20 under 35 U.S.C. § 103(a)

The Examiner has rejected claims 1-4, 11-13, and 20-30 under 35 U.S.C. § 103(a) based on the teachings of Augusteijn, et al. (U.S. Patent 6,292,883) in view of Saylor, et al., (U.S. Patent 6,501,832).

Applicants respectfully traverse the rejection of claim 1 because neither of the cited references and, more particularly, the primary cited reference (Augusteijn) teach or suggest the limitations of claim 1. For example, claim 1 recites "a cache which stores the compiled document data prior to receipt of audio input from a given user requesting a text-based document." None of the cited references suggest this limitation, especially in the context of an interactive voice response system that serves documents in response to user's audio requests as recited by the balance of the claim.

First, neither of the cited references individually teaches or suggests pre-compiling documents that can be requested by a user of a respective interactive voice response system. The office action indicates that Augusteijn is directed towards a method of converting virtual machine instructions into "native" instructions via use of a fetcher and pre-processor. This is the main basis presented in the office action to reject claim 1. As discussed in Augusteijn, the virtual machine instructions are converted "on the fly" and are fed to a

microcontroller for execution. At column 1, lines 48-50, Augusteijn discusses a motivation for using a pre-preprocessor to perform the conversion from virtual machine instructions to "native" instructions understood by a core processor:

> "It is a further object of the invention to provide a method and processing unit of the kind set forth wherein the program is represented in a more compact form."

That is, Augusteijn discusses storing the unexecutable virtual machine code rather than the compiled executable code associated with a program because the executable version of virtual machine code is too large in size. Augusteijn describes a way to avoid having to store pre-compiled program code. For example, Augusteijn discloses conversion of non-executable code in a source program "on the fly" as mentioned above, not storage of compiled executable code. This is further discussed in Augusteijn at column 3, lines 6-20:

> "The defined conversion means is represented in the processing unit. In this way, fast execution of the program is maintained, whereas at the same time a compact representation is achieved. This method is particularly suitable for use with embedded applications. In this case, the source program relates to all program statements initially represented in the embedded system (for instance, the program present in the system when the user purchases the system). The program may be stored in a permanent memory, such as ROM, or stored in a reprogrammable memory, such as EEPROM. For embedded applications it is highly desired that the code used to represent the embedded application program is compact and that the performance in executing the code is good." (emphasis added)

Further references to this technique include the abstract of Augusteijn, which states:

> "In a pre-processing step, for the program statements of the source program a program-specific virtual machine is defined with a corresponding set of virtual machine instructions, such that the expression of the program statements in the sequence of instructions requires less storage space compared to using only native instructions." (emphasis added)

In other words, rather than store pre-compiled executable code or native instructions as is done in claim 1, Augusteijn teaches a technique of converting program statements at run-time into executable code. This technique in Augusteijn is opposite to the limitations in recited claim 1, which include storing compiled document in executable form prior to a user requesting a respective document. Thus, Augusteijn teaches away from the claimed invention.

Moreover, Augusteijn does not store compiled document data as described in the context of an interactive voice response system as in the present invention. One of ordinary skill in the art would interpret a "document" in the claimed invention to be a resource such as a text-based document for maintaining response data potentially requested by a user of the interactive voice response system. Neither Augusteijn nor Saylor discuss pre-compiling such documents prior to execution of a user requesting a respective text-based document. Instead, Augusteijn discusses executing an application program or source code. There is no mention of a document that can be requested from a user via audio input. Thus, even a combination of references does not recite the limitations as in claim 1.

As discussed in the last office action reply, a benefit of storing pre-compiled executable document data as in the claimed invention is the ability of the claimed interactive voice response system to respond more quickly to a user requesting the text-based document because the document data is already in compiled form and stored in the cache prior to the request.

For the reasons stated in the last reply, Applicants submit that the individual Saylor reference also does not teach or suggest storing compiled document data as described in the context of the claimed invention. Instead, Saylor "compiles" raw Vpages when they are retrieved from disk. Presumably the Examiner agrees with this analysis since the Examiner cited new art to reject the claimed invention.

Applicants submit that even a combination of techniques in Augusteijn and Saylor do not teach the claimed invention. In other words, the claimed invention includes limitations not found in either reference. For example, even if the interactive voice response system in Saylor were to convert instructions into executable code "on the fly" as in Augusteijn to support retrieval of documents of raw data such as Vpages, this is not equivalent to compiling and storing compiled documents in a cache prior to receipt of audio input from a given user requesting a respective text-based document. In other words, even if the content interpreter 66 (FIG. 3) as discussed in Saylor at column 21 lines 20-22 were to be replaced with the "on the fly" conversion of instructions as discussed in Augusteijn, there still is no indication of that the interpreter would compile "requestable" documents of the interactive voice response system for storage in a cache, especially prior to receiving an audio request from a user for a corresponding document. Thus, Applicants respectfully submit that although Vpage server 22 does include an interpreter 66, there is no mention, teaching or suggestion that 'interpreted' or 'compiled' documents (or Vpages) generated by interpreter 66 are stored or cached in memory such as database 18. Replacing the interpreter 66 according to the techniques of Augusteijn does not provide this feature either. Thus, the rejection of claim 1 is improper because the claimed invention includes a limitation not taught or suggested by either of the references.

As discussed, according to Saylor '832, a Vpage must be 'interpreted' or 'compiled' to generate an executable file (such as an audio file) each time a user requests a particular document (e.g., during a phone call). For the reasons discussed above, there is also no suggestion in Augusteijn to pre-compile documents into executable code prior to a request for such a document. In fact, Augusteijn does not even support retrieval of documents.

Applicants respectfully submit that in view of the above amendment and remarks, claim 1 is novel and non-obvious as it incorporates techniques contrary to previously accepted wisdom and blueprints for the inventive method cannot be found in the individual or combined cited references. Accordingly, Applicants submit that independent claim 1 and corresponding dependent claims 2-4 are in condition for allowance over the prior art.

For similar reasons as claim 1, claim 11 should be in condition for allowance. For example, neither Augusteijn nor Saylor teaches or suggests compiling retrieved documents. Also, contrary to the office action, Augusteijn discloses a technique of utilizing unexecutable instructions that are converted and executed on the fly. The claimed invention recites "compiling the retrieved documents into compiled document data in executable form." The cited references do not store information in executable form. Also, the claimed invention recites "caching the compiled document data for later retrieval and execution." In comparison, the Augusteijn reference discloses compiling instructions into a native format. The native executable instructions are executed immediately. Claims 12 and 13 depend from claim 11 and therefore also should be in condition for allowance.

Claim 20 includes similar limitations as discussed above for claim 1. For applicable reasons, it is submitted that independent claim 20 and corresponding dependent claims 21-29 are in condition for allowance. Note also that claim 20

includes an element not found in any of the cited references. For example, claim 20 includes "a fetcher that receives a signal from the execution thread to search a cache for executable code associated with the requested audio information." Applicants respectfully submit that there is a distinction between the fetcher in the claimed invention and that cited in Augusteijn. For example, the fetcher in Augusteijn fetches a virtual machine instruction that must be converted into a native executable code understood by the processor. The native executable code is executed immediately and is not stored for retrieval based on later requests for a respective document. In contradistinction, the fetcher in claim 20 fetches already compiled executable code associated with requested audio information. Thus, Augusteijn does not teach or suggest the claimed invention. This limitation is also not found in Saylor because Saylor also does not teach or suggest inclusion of a fetcher that retrieves already compiled executable code. Thus, claim 20 includes a limitation not found in any of the cited references and therefore is in allowable condition.

Note that claims depending from claim 20 further distinguish the claimed invention over the cited art. For example, claim 21 recites inclusion of "a compiler that converts the text-based document into executable speech code for storage in the cache prior to receipt of the incoming request." Applicants respectfully submit that neither reference includes: i) a compiler that converts text-based documents into executable code, and ii) storage of executable speech code in a cache prior to receipt of a corresponding an incoming request to the interactive voice response system. Thus, Applicants submit that claim 21 includes a limitation not found in any of the cited references and there is no basis to maintain the corresponding rejection.

Claim 22 recites inclusion of "wherein the fetcher initiates communication with a remote server to retrieve a text-based document associated with the requested information if corresponding executable code is not stored in the

cache." Applicants respectfully submit that neither reference addresses having to retrieve a text-based document not already having corresponding executable data stored in the cache. First, neither reference teaches storage of executable code associated with a text-based document. Second, neither reference teaches a technique of conditionally retrieving a document if corresponding executable code associated with the document is not already stored in the cache. Thus, Applicants submit that the claimed invention includes elements not found in either of the cited references and there is no basis to maintain the rejection of claim 21.

Claim 23 recites inclusion of "a compiler that converts retrieved text-based documents into executable code for storage in the cache." Applicants respectfully submit that neither reference includes a compiler that converts retrieved text-based documents as used in the context of an interactive voice response system into executable code for storage in a cache. Thus, Applicants submit that the claimed invention includes elements not found in either of the cited references and there is no basis to maintain the rejection of claim 23.

Claim 24 recites inclusion of "wherein executable code stored in the cache is concurrently utilized by multiple execution threads to provide a response to multiple users." Applicants respectfully submit that neither reference includes this claim limitation. For example, Augusteijn recites "on the fly" conversion of virtual machine code into executable code only for a single running processor. There is not indication that two processors would follow the exact same processing path and execute the same instructions at the same time. Moreover, as discussed above, there is no indication that of providing a response to any user, let alone multiple users. Thus, Applicants submit that the claimed invention includes elements not found in either of the cited references and there is no basis to maintain the rejection of claim 24.

Claim 26 recites "wherein the executable code retrieved from the cache is associated with a corresponding viewable text-based document available on the World Wide Web." Applicants respectfully submit that neither reference includes this claim limitation and, more particularly, the Augusteijn reference does not include the limitation of claim 26. For example, Applicants have reviewed the cited passages below at column 7, lines 1-44:

> "The instruction memory 120 contains virtual machine instructions, such as instructions for a stack machine. The instruction memory may also be used for storing data. The invention is not limited to Harvard architecture, wherein data and instructions are separate. The microcontroller 110 comprises a processor 112 with a predetermined microcontroller core 114, referred to as a native machine, for executing native instructions from a predetermined set of microcontroller specific instructions. An example of a microcontroller suitable for executing embedded software is a RISC-type microcontroller, like the MIPS PR3001 range of microprocessors. The processor may comprise an instruction cache 116 for storing native instructions before executing the instructions. The native instructions of the microcontroller core 114 are different from the virtual machine instructions of the virtual machine. As such the microcontroller 110 is not able to directly execute virtual machine instructions stored in the instruction memory 120. In response to the processor 112 requesting an instruction, the pre-processor 130 issues the native instruction. In order to be able to generate the native instruction the pre-processor 130 may fetch a virtual machine instruction from the instruction memory 120 using fetching means 134. The converter 132 of the pre-processor 130 is used for converting a virtual machine instruction, fetched from the instruction memory 120, into at least one native instruction. In general, a virtual machine instruction is converted into a sequence of native instructions. The pre-processor 130 further comprises a feeding means 136 for feeding native instructions of the sequence to the microcontroller core 114 for execution. When executing a virtual machine program the microcontroller 110 in fact executes a

native program generated by the pre-processor 130. Where normally an instruction pointer of the microcontroller 110 indicates the next instruction in the instruction memory 120 required by the microprocessor 110 to be executed next, now the instruction pointer indicates to the pre-processor 130 that a next native instruction is required (or a re-feeding of a previous instruction). Consequently, the pre-processor 130 manages an independent virtual machine instruction pointer indicating the current (or next) virtual machine instruction in the instruction memory 120. The microcontroller does not require (explicit) knowledge of the virtual machine instruction or the virtual machine instruction pointer."

There is no indication whatsoever that executable code retrieved from the cache is associated with a corresponding viewable text-based document available on the World Wide Web. Thus, Applicants submit that the claimed invention includes elements not found in either of the cited references and there is no basis to maintain the rejection of claim 26. Applicants therefore respectfully request allowance of claim 26.

Rejection of Claim 30 under 35 U.S.C. § 103

The Examiner has rejected claim 30 under 35 U.S.C. § 103(a) based on the teachings of Augusteijn, (U.S. Patent 6,292,833) and Saylor, et al, (U.S. Patent 6,501,832).

Applicants note that the office action cites art for the first two claim elements. However, there is no indication of art used to reject the last two claim elements that read as follows: "searching a cache for executable code associated with the requested audio information, the executable code generated in response to a previous request from another user for audio information associated with the text-based document; and executing corresponding executable code from the cache to satisfy the request for audio information associated with the incoming call."

Applicants respectfully submit that neither of the cited references teaches or suggests the limitations of claim 30. First, Augusteijn does not teach or suggest a technique of searching a cache for executable code associated with requested information. Second, Augusteijn does not execute code retrieved from the cache to satisfy a request for audio information associated with an incoming call. Third, and perhaps more importantly, claim 30 recites that the executable code stored in the cache is generated in response to a previous request by another user for audio information associated with the same text-based document. In other words, a first user that previously requested the same text-based document causes the document to be compiled and stored in the cache for later use by other requesting users. According to the invention, there is no need to recompile the document for the later requesting user because the compiled executable document is stored in the cache. Thus, Applicants submit that the claimed invention includes elements not found in either of the cited references and there is no basis to maintain the rejection of claim 30. Applicants therefore respectfully request allowance of claim 30.

New claims 31-39

For applicable reasons as discussed above, claim 31-39 are in condition for allowance. Applicants respectfully request allowance of these claims as well.

Conclusion

In view of the foregoing remarks, Applicants is respectfully submit that the claims of the present application are in condition for allowance. A Notice to this affect is respectfully requested. If the Examiner believes, after submission of this reply, that the Application is not in condition for allowance, the Examiner is respectfully requested to call the Applicants' Representative at the number below.

Applicants hereby petition for any extension of time which is required to maintain the pendency of this case. If there is a fee occasioned by this response, including an extension fee, that is not covered by an enclosed check, please charge any deficiency to Deposit Account No. 50-0901.

If the enclosed papers or fees are considered incomplete, the Patent Office is respectfully requested to contact the undersigned Attorney at (508) 366-9600, in Westborough, Massachusetts

Respectfully submitted,

Paul P. Kriz, Esq.
Attorney for Applicants
Registration No.: 45,752
CHAPIN & HUANG, L.L.C.
Westborough Office Park
1700 West Park Drive
Westborough, Massachusetts 01581
Telephone: (508) 366-9600
Facsimile: (508) 616-9805

Attorney Docket No.: NMS03-05

Dated: July 21, 2005